**Master’s Project Proposal**

**CS700B**

**Building Menu Data Structure for Fast Ontology Access**

Submitted to the

Department of Computer Science

College of Computing Sciences

New Jersey Institute of Technology

In Partial Fulfillment of

The Requirements for the Degree of

Master’s of Science

By

Shrutee C. Shah

ss658@njit.edu

|  |  |
| --- | --- |
| Proposal Number |  |
| Agree to Advise | **Dr. James Geller** (*Project Advisor*) |
| Date Submitted |  |
| Approved By | (*MS in CS Committee*) |
| Date Approved |  |

**Abstract**

The `Ontology-Enabled Web Search` project aims at utilizing the vast information extracted from the Deep Web in assisting users in their search for Web pages. This is done by increasing the number of relevant Web sites returned. To that end, the project aims at mining the Deep Web for useful information, building domain specific ontologies with that information and finally providing the users with a friendly interface for analyzing the search terms entered and getting relevant Web sites to show the contribution of the enriched ontology to Web search effectiveness. Mining the Deep Web for relevant instances based upon a particular ontology is an important process as this information is used in the development and enriching of the ontologies and thus making them more usable for indexing of the Deep Web sites.

My area of concentration is creating a front end of the Deep Web mining application to search for famous people that are part of the mining database, represented in the form of an ontology. I will also work on determining the right data structure or database that would give a faster response time for suggestions when users type in data. Currently, the Deep Web is being mined for ‘Politicians’, ‘Religious Leaders’, ‘Researchers’ [15] and in general for famous people [14]. Previously, the mining of the Deep Web for ‘singers’ and ‘basketball players’ was done by mining on a particular website and then gathering relevant information from it [12]. Before that the mining of the Deep Web for the `Airports` ontology was done by implementing a parser that gathers information from a Deep Web site by filling out the surface forms depending on the structure of that particular Web page [2].

I will be developing a Graphical User Interface to search Information about famous people using `Ontology-enabled Web search.` This Interface will be connected to the ontology, and will parse user queries and display relevant results in a categorized manner. I will also implement a data structure for fast access of the ontology in order to reduce the response time for each search.

**Table of Contents**

Abstract....................................................................................................................................3

1. Introduction...…………………..………………………………………………………...5
   1. What is Deep Web……….………………………………………………………..…5
   2. Ontologies……………….………………………………………………………...…6
   3. Deep Web Mining……….………………………………………………………...…8
   4. Role of Ontologies in search for Web pages………………………………………..11
2. Previous Work…………………………………………………………………………...13

2.1 Enriching Ontology for Deep Web Search…………………………………………..13

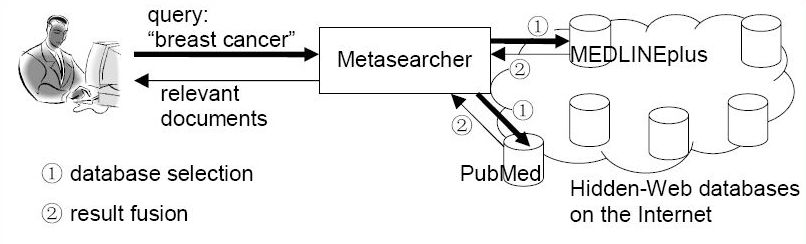
2.2 Trie Data Structure………………………………………………...………………...14

2.2.1 Searching a Trie……………………………..…………………………………15

2.2.2 Inserting into a Trie....……………………………..…………………………...15

1. My Work…………………………………………………………………………….…...16
   1. Scenario………………………………………………………………………….…...16
   2. Methodology…………………………………………………………………….…...16
   3. Deliverables……………………………………………....………………………….19
   4. Weekly Plan………………………………………………………………………….21
2. **Introduction**
   1. **What is the Deep Web?**

The DeepWeb refers to information served up on websites that is hidden or generally inaccessible through traditional search methods as these results are not indexed by search engines. For example, information that resides or is provided through searchable databases, the results of which can only be discovered through queries or by filling out Web based forms is considered Deep Web data. It contains web pages that are dynamically generated via query interfaces implemented as web forms or web services it is not part of the [surface web](http://en.wikipedia.org/wiki/Surface_Web). A great deal of information may be caught in the web crawlers and search engines, but there is a wealth of information that is deep and therefore missed. Most of the web's information is buried far down on dynamically generated sites, and standard search engines do not find it. These web pages do not exist until they are created dynamically as the result of a specific search [12]. The figure1 below shows one method of crawling Deep Web [11].



**Figure 1:** Web Crawling Method [11].

Due to its dynamic nature, existing Web crawlers cannot access the Deep Web. Thus, accessing and maintaining the huge amount of Deep Web information remain challenging research issues. Information in Deep Web sites Information is categorized as being either in textual or structured databases. While a textual database needs input keywords for searching text documents, a structured database requires a user to fill in input fields of a query interface [12].

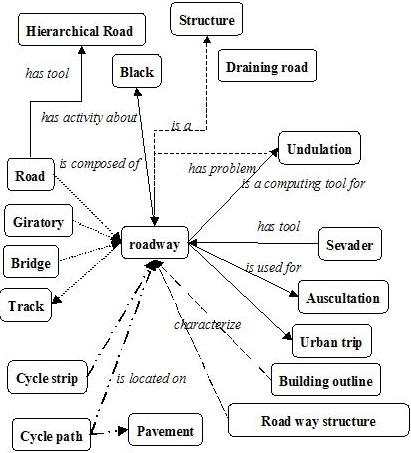
Deep Web content includes information in private databases that are accessible over the Internet but search engines are unable to crawl due to various reasons. For example, some universities, government agencies and other organizations maintain databases of information that were not created for general public access. Other sites may restrict database access to members or subscribers [12].

The Deep Web contained somewhere in the vicinity of 900 billion pages of information. In contrast, Google, the largest search engine, had indexed just 25 billion pages [8].The term, "Deep Web," was coined by BrightPlanet, an Internet search technology company that specializes in searching deep Web content. In their 2001 white paper, 'The Deep Web: Surfacing Hidden Value,' BrightPlanet noted that the Deep Web was growing much more quickly than the Surface Web and that the quality of the content within it was significantly higher than the vast majority of Surface Web content. Although some of the content is not open to the general public, BrightPlanet estimates that 95% of the Deep Web can be accessed through specialized search [9].

**1.2 Ontologies**

Ontology in [computer science](http://en.wikipedia.org/wiki/Computer_science) and [information science](http://en.wikipedia.org/wiki/Information_science) is a formal representation of a set of concepts within a [domain](http://en.wikipedia.org/wiki/Domain_of_discourse) and the relationships between those concepts. It is used to [reason](http://en.wikipedia.org/wiki/Reasoning) about the properties of that domain, and may be used to define the domain. In the words of Thomas Gruber ontology is an explicit specification of a conceptualization. A conceptualization is an abstract, simplified view of the world that we want to represent. If the specification medium is a formal language, the ontology defines a representational foundation [2].

Example: Figure 2 gives a pictorial representation of roadway ontology which consists of various instances and there corresponding relations.



**Figure 2:** An example of roadway ontology [10].

According to Dr. James Geller’s more precise and detailed definition of ontology it is a graph (the data Structure). Every Node of this graph stands for a “concept” which is a unit that one can think about and corresponds to words or short phrases. Typically, a concept corresponds to a noun or noun phrase like house, man, car, New York etc. but that is not an obligation [1].

The nodes of the ontology are connected by different kinds of links. The most important kind of link is called IS-A link. The nodes and IS-A links together form a Rooted Directed Acyclic Graph (Rooted DAG).Rooted means that there is one single "highest node" called the Root. All other nodes are connected by one IS-A link or a chain of several IS-A links to the Root. In our definition, IS-A links point upward. If an IS-A link points from a concept X to a concept Y that means that every real world thing that can be called an X also can be called a Y. In other words, every X IS-A Y. (Some people have IS-A-like links but pointing downwards.) Examples: A car IS-A vehicle. A dog IS-A animal [1].

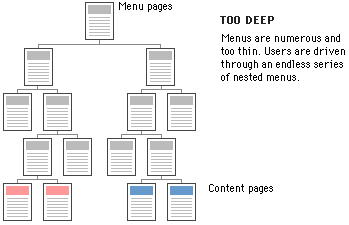
Acyclic means that if you start at one node and move away from it following an IS-A link, you can never return to this node, even if you follow many IS-A links. Most nodes also have other information attached. This information includes attributes, relationships and rules [1].

Ontology represents information in a form that can be used for some forms of reasoning that is at least partially similar to human reasoning. This includes inheritance reasoning, transitivity reasoning and classification. A concept may inherit information from several other concepts. This is called multiple inheritance. Transitivity reasoning corresponds to chaining of IS-A links. Classification means that if we know the attributes of a concept we can decide under which other concepts it belongs in the ontology [12].

**1.3 Deep Web Mining**

Just because a Web search engine can't find something doesn't mean it isn't there. The Deep Web is a vast information repository not always indexed by automated search engines but readily accessible to enlightened individuals. The Shallow Web, also known as the Surface Web or Static Web, is a collection of Web sites indexed by automated search engines [12].

A search engine robots or Web crawler follows URL links, indexes the content and then relays the results back to search engine central for consolidation and user query. Ideally, the process eventually scours the entire Web, subject to vendor time and storage constraints. Most of the time information is stored on Web sites in such a way that the user initially comes in contact with what are called Menu pages. Menus are numerous and too thin i.e. they are having basic information and users are driven through an endless series of nested menus in order to reach important information stored on backend inside Content pages as shown in Figure 2 below [12].



**Figure 3:** Information storage format on a typical website [12].

The crux of the process lies in the indexing. A Web crawler does not report what it can't index.

And we know the search result for a particular Web page in terms of its relevance depends greatly on that. This was a minor issue when the early Web consisted primarily of static generic HTML code, but contemporary Web sites now contain multimedia, scripts and other forms of dynamic content. The Deep Web consists of Web pages that search engines cannot or will not index. The popular term "Invisible Web" which refers to Deep Web is actually a misnomer, because the information is not invisible, it's just not indexed by the Web crawler. The Deep Web is five to 500 times as vast as the Shallow Web, thus making it an immense and extraordinary online resource. The major search engines together index approximately 20% of the Web, and thus missing 80% of the content [12].

Search engines typically do not index the following types of Web sites:

* Proprietary sites
* Sites requiring a registration
* Sites with scripts
* Dynamic sites
* Ephemeral sites
* Sites blocked by local webmasters
* Sites blocked by search engine policy
* Sites with special formats
* Searchable databases

Proprietary sites require a fee. Registration sites require a login or password. A Web crawler can index script code (e.g., Flash, JavaScript), but it can't always ascertain what the script actually does and the Web crawler may get trapped within infinite loops. Dynamic Web sites are created on demand and have no existence prior to the query and limited existence afterward [12].

Webmasters can request that their sites not be indexed (Robot Exclusion Protocol), and some search engines skip sites based on their own inscrutable corporate policies. Not long ago, search engines could not index files in PDF, thus missing an enormous quantity of vendor white papers and technical reports, not to mention government documents. Special formats become less of an issue as index engines become smarter. The most valuable Deep Web resources are searchable databases. There are thousands of high-quality, authoritative online specialty databases. These resources are extremely useful for a focused search [12].

**1.4 Role of Ontology in search for Web pages**

Recently, there has been a growing interest in Web searches that are intended to locate information that exists in the backend data bases of Web services. Web sites in E-commerce domains such as airfares, automobiles, books, car rentals, hotels, jobs, movies and music records usually store huge amounts of information, which is of interest to many users, in their backend databases [12].

Information in E-commerce backend databases is usually not “visible” to general search engines. The information in backend databases is often called Deep Web data. Finding the relevant E-commerce sites and accessing, retrieving and indexing the huge amounts of Deep Web data raises challenging research issues [12].

Ontology could play an important role in assisting users in their search for Web pages. Domain ontology can be constructed that support users in their Web search efforts and that increase the number of relevant Web pages that are returned. To achieve this goal the Deep Web information, which consists of dynamically generated Web pages, which cannot be indexed by the existing automated Web crawlers, is combined with ontology[12].

The process of building ontology consists of several steps as shown in Figure 4. Firstly the possible search terms, called attributes of Deep Web data sources are automatically extracted from a static collection of Deep Web sites. Secondly, separate domain ontology is built for each domain, using the extracted search terms. Thirdly, by probing a few Deep Web sites domain terms from the backend databases are extracted. Next, the domain ontology is extended to include these Deep Web terms as instances. Finally, the domain ontology is extended with relationships between instances [3].



**Figure 4:** The Flow for an Ontology-Supported Deep Web Search [3].

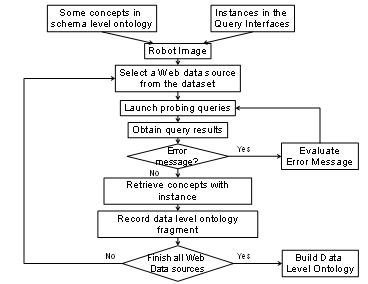
**2. Previous Work**

The previous work on this project included addressing the problems of extracting instances from the Deep Web, enriching a domain specific ontology with those instances, and using this ontology to improve web search. In the initial phase of the project information was gathered about all the airports located all across North America by mining the Deep Web for building an `Airports` ontology based on that. This information was then used to provide users with additional search terms for making the search more specific and getting more relevant web pages. The effectiveness of this methodology was shown by comparing the number of relevant Websites by a search engine with a user’s search terms only, with the websites found when using additional ontology-based search terms. In the next phase of the project the domain was extended to searching out information about famous people like singers, publishers, actors, actresses, basketball players, researchers, religious leaders etc [14][15]. Current phase involves building of widely used Data structure called Trie, to enable faster search of Onotlogies.

**2.1 Enriching Ontology for Deep Web Search**

Enrichment of ontology is a process that extends it by adding concepts, instances and new relations between concepts. In the related paper method for extracting instances from the DW is based on developing “robots” (agents) that sent many queries to the same DW site to extract as many data values as possible. When a robot encounters an input field it may enter random values or leave the field empty and then submit the page to elicit an informative response [3].

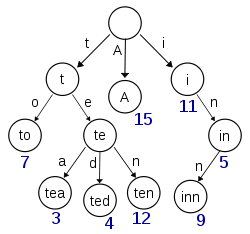
Figure 5 shows the workflow for extending the ontology with instances. The concept discovery of the robot is guided by a human in its initial stage. Initial pairs of a concept and its corresponding instances are defined, which we call a *robot image*. The robot submits input values into the query interface. If the input values are not suitable for the form, most Web sites display error messages. The analysis of the error messages often gives useful clues to the robot to guess suitable input values and launch better probing queries. Thus, the queried Web sources may provide information about concepts, instances and semantic relationships, which is recorded in the ontology [3].



**Figure 5:** A Flow for generating data level ontology fragments [3].

**2.2 Trie Data Strucuture**

A **trie**, or prefix tree, is an [ordered tree](http://en.wikipedia.org/wiki/Ordered_tree_data_structure) [data structure](http://en.wikipedia.org/wiki/Data_structure) that is used to store an [associative array](http://en.wikipedia.org/wiki/Associative_array) where the keys are usually [strings](http://en.wikipedia.org/wiki/String_(computer_science)). Unlike a [binary search tree](http://en.wikipedia.org/wiki/Binary_search_tree), no node in the tree stores the key associated with that node; instead, its position in the tree shows what key it is associated with. All the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the [empty string](http://en.wikipedia.org/wiki/String_(computer_science)). Values are normally not associated with every node, only with leaves and some inner nodes that correspond to keys of interest. Trie comes from word ‘retrieval’ [16].



**Figure 6:** Example of Trie with Keys A, to, tea, ted, ten, in, inn [16]

* + 1. **Searching a Trie**

To search a trie for an element with a given key, we start at the root and follow a path down the trie until we either fall off the trie (i.e., we follow a null pointer in a branch node) or we reach an element node. The path we follow is determined by the letters of the search key [13].

Consider figure 6; suppose we want to search for ‘tea.’ Then we start from the root node, and follow the path in the trie, character by character. From the root, we go to ‘t,’ then ‘e’ and then following the path for ‘a’ we find the keyword ‘tea.’ Now, suppose we want to search for ‘tax.’ Then, we start from the root node towards ‘t,’ but there is no branch for ‘a’ from ‘t,’ hence we know that there is no keyword ‘tax’ in the given trie.

* + 1. **Inserting into a Trie**

To insert an element x whose key is y, we first search the trie for an existing element with this key. If the trie contains such an element, then we replace the existing element with x. When the trie contains no element whose key equals y, x is inserted into the trie.

1. **My Work**

**3.1 Scenario**

The target of this project is to mine instances from the Deep Web, and then enrich a domain specific ontology with those instances, and finally use this ontology to improve Web search performance for users. My work is an extension of the Deep Web Mining project where in I will develop the front end of the search tool, where users can type information or names of famous people and the graphical user interface should give suggestions, for now about famous personalities and their various attributes, changing after each typed letter, based on the indexed names the system has in the ontology or backend database.

I will also be working on developing methods for faster indexing of the backend database or creating data structures or algorithms that would facilitate providing relevant suggestions and faster search results to users.

**3.2 Methodology**

I will develop a web-based form as well as a standalone Graphical User Interface for extracting information from the backend database. This database has stored information of famous celebrities in various fields, derived by Deep Web mining of several Web sites. The first step would be to determine the right kind of medium, i.e., a data structure or database for faster indexing and providing quick output of relevant entries. I will test both the media for their performance and will select the one with the better performance. The next step would be to develop an algorithm to utilize this medium to provide quick suggestions to users.

The next step involves creating the Graphical User Interface (GUI) which will interact with the `ontology-enabled database` in the backend. It will be a very lightweight GUI with a textbox for a user to enter his search queries. When the user starts typing, for each character, matching suggestions are made. The number of suggestions will be limited to ten and could be extended to fifteen in case there is more than one disjoint result to be displayed separated by a line.

The work flow for this process is also depicted in Figure 8.

Ontology-enabled Database

**Deep Web**

Data Structure to get data from the database

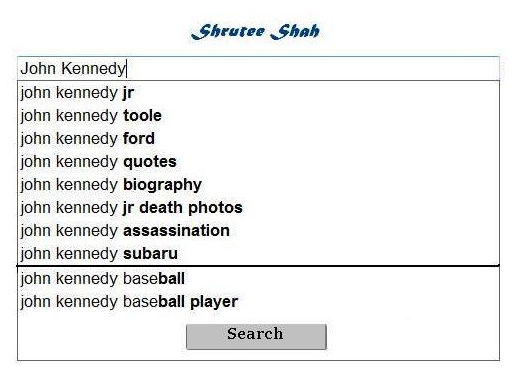
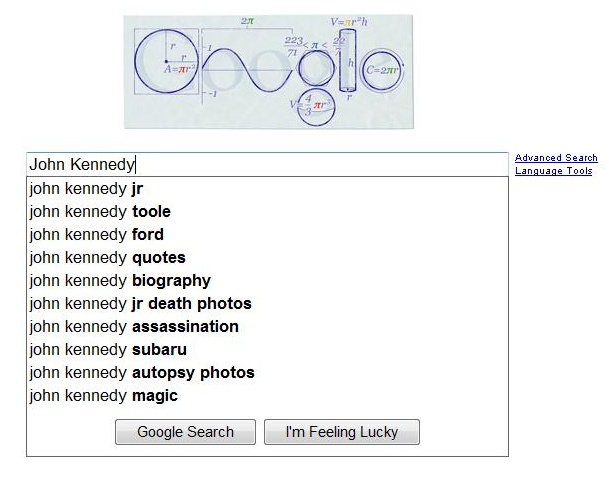
Form and standalone application to display results

**Figure 8:** Flow of Data to the stand-alone application.

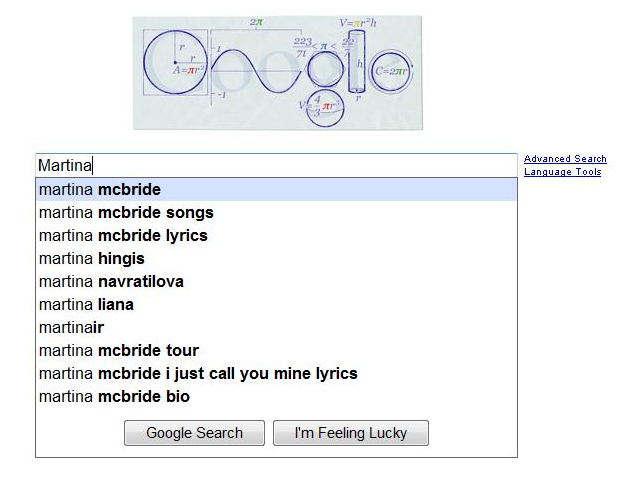
Figure 9a and 9b show difference in two search engines, the most widely used search engine `Google` and my application which uses `an ontology-enabled database` for search. We have two examples here. In Figure 9a, we see that suggestions given by Google are related only to *President* John Kennedy, whereas my `ontology-enabled database` will give suggestion relevant to the *baseball player* John Kennedy too. Thus, it reduces the effort of typing complete phrases such as ‘John Kennedy Baseball.’

Figure 9b shows the comparison for searching ‘Martina’ with Google and with my interface. We get mixed search suggestions relevant to all categories from Google, while my system categorizes suggestions. As can be seen, I will have two levels of categorization. ‘Martina McBride’ falls into the singer category, and there are two ‘Martinas’ in the tennis category – ‘Martina Hingis’ and ‘Martina Navratilova.’ This categorization makes it easy for a user to spot the category or person they are looking for.

Note that the horizontal lines separating different suggestions in my system are of different thickness. The line separating singers from tennis players is heavier than the line separating two tennis players from each other. This is intentional to better guide the user.



**Figure 9a:** Comparison between Google and `Onotology enabled Web-Search` for search term ‘John Kennedy’ [17]



**Figure 9b:** Comparison between `Google` and `onotology enabled Web-search` for search term ‘Martina’ [17]

The limitation of this process is that the results shown will be limited to the famous people stored in the database and the results will become more useful as the database grows bigger. In case that no relevant results are found, the output will be a simple message shown on the form. Also the authenticity of the results is dependent on the authenticity of the data available in the database.

My frontend will be used to make it easier for the user to determine the actual search terms. The number of letters typed should be minimized. However, the actual search will be performed by Google by passing the selected search terms back to it.

**3.3 Deliverables**

1. **Database Indexing**

**Input:** Database or Data structure with information about famous people

**Process:** Writing queries to index the database on the columns which are used most frequently to allow fast search

**Output:** Indexed Database

**2. Trie Data Structure building**

**Input:** Database or data structure with information about famous personalities

**Process:** Writing an algorithm to build Trie from the database of famous personalities

**Output:** Trie data structure with all information about famous people extracted from `ontology-enabled Database`

**3. Medium Comparison and Analysis**

**Input:** Indexed database and data structure with information of famous people

**Process:**

* Manually write search queries for searching indexed database and analyze performance
* Manually write search queries to search information about famous people from Trie structure and analyze performance
* Compare performance of both the media based on response time and relevance to entered search term

**Output:** Determined storage medium for ontology

**4. Graphical User Interface for searching famous people**

**Input:** Ontology

**Process:**

* Develop User Interface which is connected to ontology on the backend

**Output:** Display the suggestions like a menu below the search box with disjoint searches separated by a line. Extract and display relevant results for search term entered by user

**3.4 Weekly Plan**

|  |  |
| --- | --- |
| Week 1 | Getting Started: Studying work done previously, the limitations and problems faced previously. |
| Week 2 | Researching Data Structure and Database Indexing techniques which will help getting results faster. |
| Week 3 | Continued |
| Week 4 | Implementing the Database Indexing and Writing the code to implement Trie data structure |
| Week 5 | Continued |
| Week 6 | Implement Trie data structure using the code. Querying the Trie and analyzing the performance |
| Week 7 | Querying the ‘indexed database’ and carrying out analysis of Database performance |
| Week 8 | Comparison and Analysis of Indexed Database and Trie performances |
| Week 9 | Writing code for Graphical User Interface to search famous people |
| Week 10 | Continued |
| Week 11 | Continued |
| Week 12 | Testing the Graphical User Interface for searching Ontology |
| Week 13 | Continued. |
| Week 14 | Writing Report for the work done. |

**References**

1. James Geller. Definition of Ontology,

<http://web.njit.edu/~geller/what_is_an_ontology.html>, Accessed February 20, 2010.

1. **Thomas Gruber (2008),** [**Ontology**](http://tomgruber.org/writing/ontology-in-encyclopedia.htm)**. Entry in the**Encyclopedia of Database Systems***,* Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag. February 20, 2010.**
2. Yoo Jung An, [Soon Ae Chun](http://www.informatik.uni-trier.de/%7Eley/db/indices/a-tree/c/Chun:Soon_Ae.html), [Kuo-chuan Huang](http://www.informatik.uni-trier.de/%7Eley/db/indices/a-tree/h/Huang:Kuo=chuan.html), [James Geller](http://www.informatik.uni-trier.de/%7Eley/db/indices/a-tree/g/Geller:James.html): Enriching Ontology for Deep Web Search. [DEXA 2008](http://www.informatik.uni-trier.de/%7Eley/db/conf/dexa/dexa2008.html#AnCHG08): 73-80, February 26, 2010.
3. Yoo Jung An, James Geller, Yi-Ta Wu, Soon Ae Chun: Semantic Deep Web-Automatic Attribute Extraction from the Deep Web Data Sources, March 2, 2010.
4. Yoo Jung An, Soon Ae Chun, Kuo-chuan Huang, James Geller: Assessment for Ontology Supported Deep Web Search, March 7, 2010.
5. Yagnesh Bhat, Building Ontologies for Ontology Enabled Web Search, NJIT, CS Department, MS Proposal Spring 2009.
6. Tian Tian,Enriching Multi-Domain Ontology for Web Search, PhD Project Proposal, NJIT, CS Department, in preparation of Ph.D.
7. <http://en.wikipedia.org/wiki/Google_search>, Accessed February 27, 2010.
8. <http://brightplanet.com>, Accessed February 27, 2010.
9. <http://www.cybergeo.eu/docannexe/image/8322/img-1.jpg>, Accessed March 12, 2010.
10. Frontiers in Web Data Management,<http://oak.cs.ucla.edu/~cho/papers/cho-icts03.pdf>, Accessed March 12, 2010.
11. Jalaj Asher, Creating a Database from Deep Web Sources, NJIT, CS Department, MS Proposal, Fall 2010.
12. <http://www.di.unipi.it/didadoc/asw/grossidir/Sahni/tries.htm>, Accessed March 14, 2010.
13. Mansi Pedgaonkar, Deep Web Mining for Information Retrieval about Famous People in Publishing and Acting, NJIT, CS Department, MS Proposal Fall 10.
14. Anushri Mahajan, Building Ontologies of Politicians, Religious Leaders, Researchers for in more Accurate Web Searches, NJIT, CS Department, MS Proposal Fall 10.
15. <http://en.wikipedia.org/wiki/Trie>, Accessed March 15, 2010.
16. <http://www.google.com/webhp?rls=ig>, Accessed March 14, 2010.